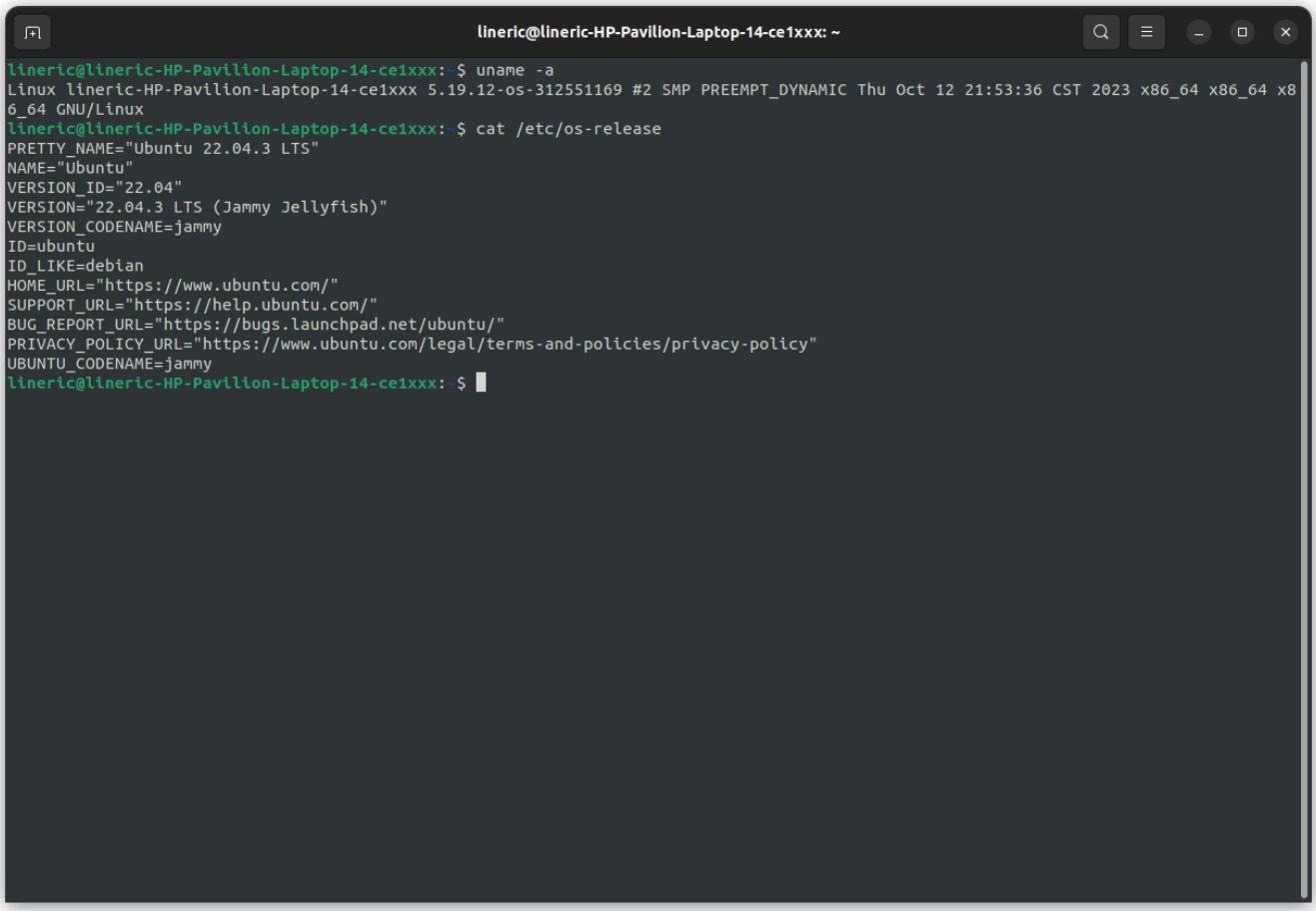
Assignment 1

Kernel Compilation



System Call

We first implement a `sys_hello` system call which show `Hello World!` and my id `312551169`.

1. Download Kernel Source Code (Here use 5.19.12)
2. Download some essential tools

```
sudo apt-get install build-dep linux libncurses-dev gawk flex bison
openssl libssl-dev dkms libelf-dev libudev-dev libpci-dev libiberty-dev
autoconf llvm
```

3. Decompress the download file and cd into it

```
tar xvf linux-5.16.16.tar.xz
cd linux-5.16.16
```

4. make a new directory(here we create hello) and create a new c file

```
mkdir hello
cd hello
touch hello.c
```

5. c file

```
#include <linux/kernel.h>
asmlinkage long __x64_sys_hello(void){
    printk(KERN_INFO "Hello world\n");
    printk(KERN_INFO "312551169\n");
    return 0;
}
```

> `asmlinkage` is a #define for some gcc magic that tells the compiler that the function should not expect to find any of its arguments in registers (a common optimization), but only on the CPU's stack

> For compatibility between 32- and 64-bit systems, system calls defined to return an int in user-space return a long in the kernel.

> `printk` print to kernel log buffer

> `KERN_INFO` is log level

6. create a Makefile

```
touch Makefile
```

7. Makefile

```
obj-y := hello.o
```

> `obj-y` compiled into the kernel or module. `obj-m` compiled into loadable kernel modules

8. go back to linux-5.19.12 directory and edit it Makefile

```
cd ..
vim Makefile
```

9. find `core-y := ...` line and add your directory in there(which is `hello` here)

```
core-y          := init/ usr/ arch/$(SRCARCH)/ hello/
```

> `core-y` is source directory in default Makefile.

10. edit `include/linux/syscalls.h` and add this line in the file

```
asmlinkage long __x64_sys_hello(void);
```

> `include/linux/syscalls.h` is a header file that defines the system call interface

11. edit `arch/x86/entry/syscalls/syscall_64.tbl` and add this line in the file

```
451     common  hello                   sys_hello
```

> The first column is the system call's number
> The second column says that this system call is common to both 32-bit and 64-bit CPUs.
> The third column is the name of the system call.
> The fourth is the name of the function implementing it.

> 451 number can be change, just don't use already used number

> After linux4.17 will auto add *_x64* at the front of syscall

12. use `make menuconfig` to create a new `.config` file
13. use `make -j$(nproc)` to compile the kernel

> prior to 2.6, you need `make modules`, while after 2.6, `make` will also do `make modules`

14. use `make modules_install -j$(nproc)` to install modules

> `make modules_install` will make sure that there are compiled binaries (and compile the modules, if not) and install the binaries into your kernel's modules directory.

15. use `make install` to install kernel to your computer

> `make install` will auto run `update-grub`

16. reboot and select new kernel
17. test it with c file

```
#include <assert.h>
#include <unistd.h>
#include <sys/syscall.h>

/*
 * You must copy the __NR_hello marco from
 * <your-kernel-build-
dir>/arch/x86/include/generated/uapi/asam/unistd_64.h
 * In this example, the value of __NR_hello is 451
 */
#define __NR_hello 451
```

```
int main(int argc, char *argv[]) {
    int ret = syscall(__NR_hello);
    assert(ret == 0);

    return 0;
}
```

18. use `dmesg` to see the kernel log

And then we implement a `sys_revstr` system call which reverse the string you type in.

- step 1~3 is the same as above.

4. make a new directory(here we create revstr) and create a new c file

```
mkdir revstr
cd revstr
touch revstr.c
```

5. c file

```
#include <linux/syscalls.h>
#include <linux/kernel.h>
#include <linux/slab.h>

SYSCALL_DEFINE2(revstr, int, length, char __user *, str) {

    char *buffer = kmalloc(sizeof(char) * (length+1), GFP_KERNEL);
    int i, j;
    unsigned long len = length;

    if (copy_from_user(buffer, str, len+1)) {
        return -EFAULT;
    }

     printk(KERN_INFO "The origin string: %s\n", buffer);

    for (i = 0, j = length - 1; i < j; i++, j--) {
        char temp = buffer[i];
        buffer[i] = buffer[j];
        buffer[j] = temp;
    }

    printk(KERN_INFO "The reversed string: %s\n", buffer);

    kfree(buffer);

    return 0;
}
```

> `SYSCALL_DEFINEn` is a set of macros that are used to define the implementation of system calls. `n` is the number of arguments. here we use 2.

> `copy_from_user(to, from, n)` is to copy data **from** user space **to** kernel space with **n** bytes of data.

> `kmalloc`, `kfree` is like kernel's `malloc` and `free`

> `GFP_KERNEL` is the flag control the allocators behavior.

6. create a Makefile

```
touch Makefile
```

7. Makefile

```
obj-y := revstr.o
```

8. go back to linux-5.19.12 directory and edit it Makefile

```
cd ..
vim Makefile
```

9. find `core-y := ...` line and add your directory in there(which is `revstr` here)

```
core-y          := init/ usr/ arch/$(SRCARCH)/ hello/ revstr/
```

10. edit `include/linux/syscalls.h` and add this line in the file

```
asmlinkage long __x64_sys_revstr(int length, char __user *str);
```

11. edit `arch/x86/entry/syscalls/syscall_64.tbl` and add this line in the file

```
452     common  revstr                  sys_revstr
```

12. use `make menuconfig` to create a new `.config` file
13. use `make -j$(nproc)` to compile the kernel
14. use `make modules_install -j$(nproc)` to install modules
15. use `make install` to install kernel to your computer
16. reboot and select new kernel

17. test it with c file

```
#include <assert.h>
#include <unistd.h>
#include <sys/syscall.h>
/*
 * You must copy the __NR_revstr marco from
 * <your-kernel-build-
dir>/arch/x86/include/generated/uapi/asam/unistd_64.h
 * In this example, the value of __NR_revstr is 452
 */
#define __NR_revstr 452

int main(int argc, char *argv[]) {
    int ret1 = syscall(__NR_revstr, 5, "hello");
    assert(ret1 == 0);

    int ret2 = syscall(__NR_revstr, 11, "5Y573M C411");
    assert(ret2 == 0);

    return 0;
}
```

18. use `dmesg` to see the kernel log

```
lineric@lineric:~/Downloads
ation="sendmsg" profile="/snap/snapd/20290/usr/lib/snapd/snap-confine" pid=1275
comm="snap-confine" family="unix" sock_type="stream" protocol=0 requested_mask="
send" denied_mask="send"
[   27.045726] rfkill: input handler disabled
[   47.515786] audit: type=1400 audit(1697520675.080:65): apparmor="DENIED" oper
ation="capable" profile="/snap/snapd/20290/usr/lib/snapd/snap-confine" pid=1817
comm="snap-confine" capability=12  capname="net_admin"
[   47.515791] audit: type=1400 audit(1697520675.080:66): apparmor="DENIED" oper
ation="capable" profile="/snap/snapd/20290/usr/lib/snapd/snap-confine" pid=1817
comm="snap-confine" capability=38  capname="perfmon"
[   47.556718] Bluetooth: RFCOMM TTY layer initialized
[   47.556722] Bluetooth: RFCOMM socket layer initialized
[   47.556725] Bluetooth: RFCOMM ver 1.11
[   47.900660] rfkill: input handler enabled
[   48.903681] rfkill: input handler disabled
[   49.088172] logitech-hidpp-device 0003:046D:4074.0007: HID++ 4.2 device conne
cted.
[   59.824415] Hello world
[   59.824420] 312551169
[   70.847737] The origin string: hello
[   70.847744] The reversed string: olleh
[   70.847746] The origin string: 5Y573M C411
[   70.847748] The reversed string: 114C M375Y5
➜  Downloads
```